

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY  
BANGALORE

SUBJECT VLSI ARCHITECTURE DESIGN  
COURSECODE VL 731

---

## Project

---

*Yathin Kumar*  
(IMT2020550)

*B Sathiya Naraayanan*  
(IMT2020534)

*Maurya Patel*  
(IMT2020517)

*Paras Vekariya*  
(IMT2020547)

March 14, 2023

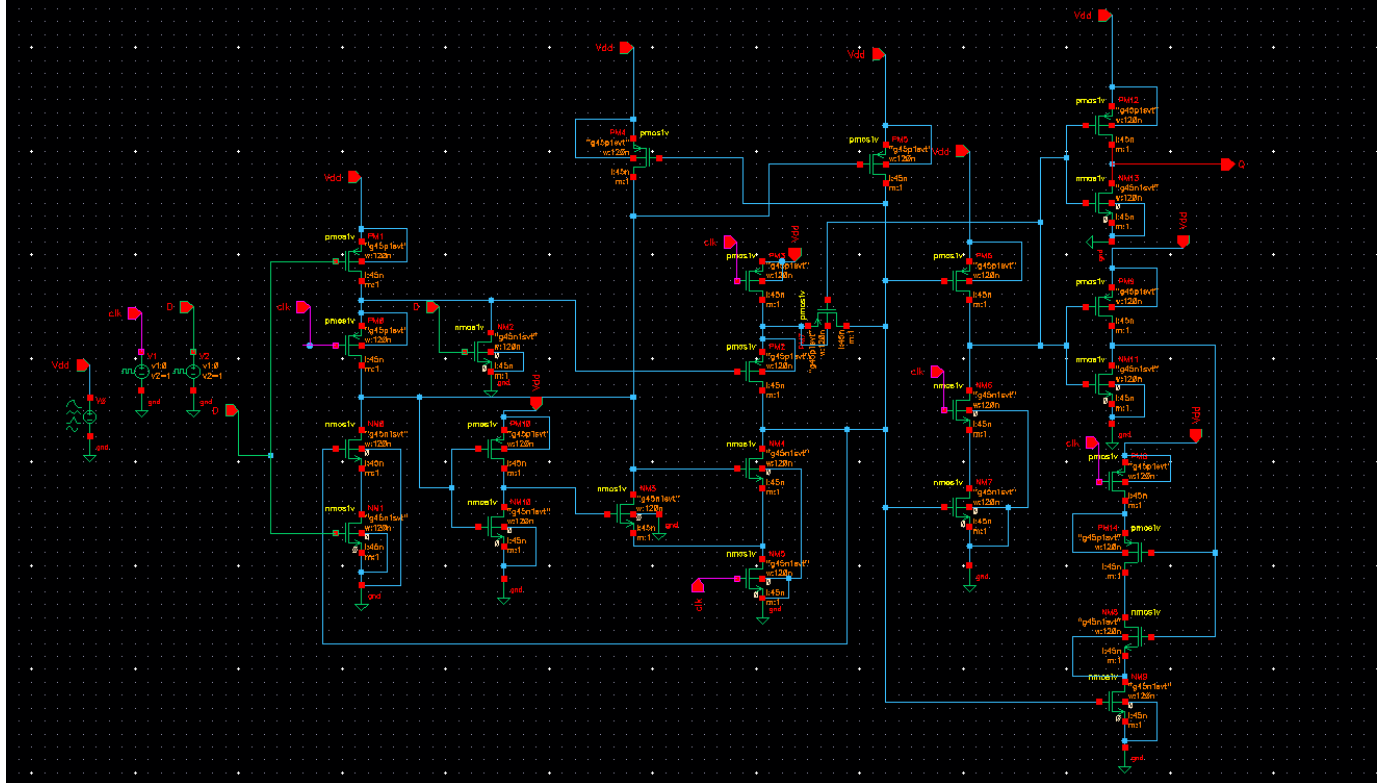


# 1 PROPOSED ARCHITECTURE

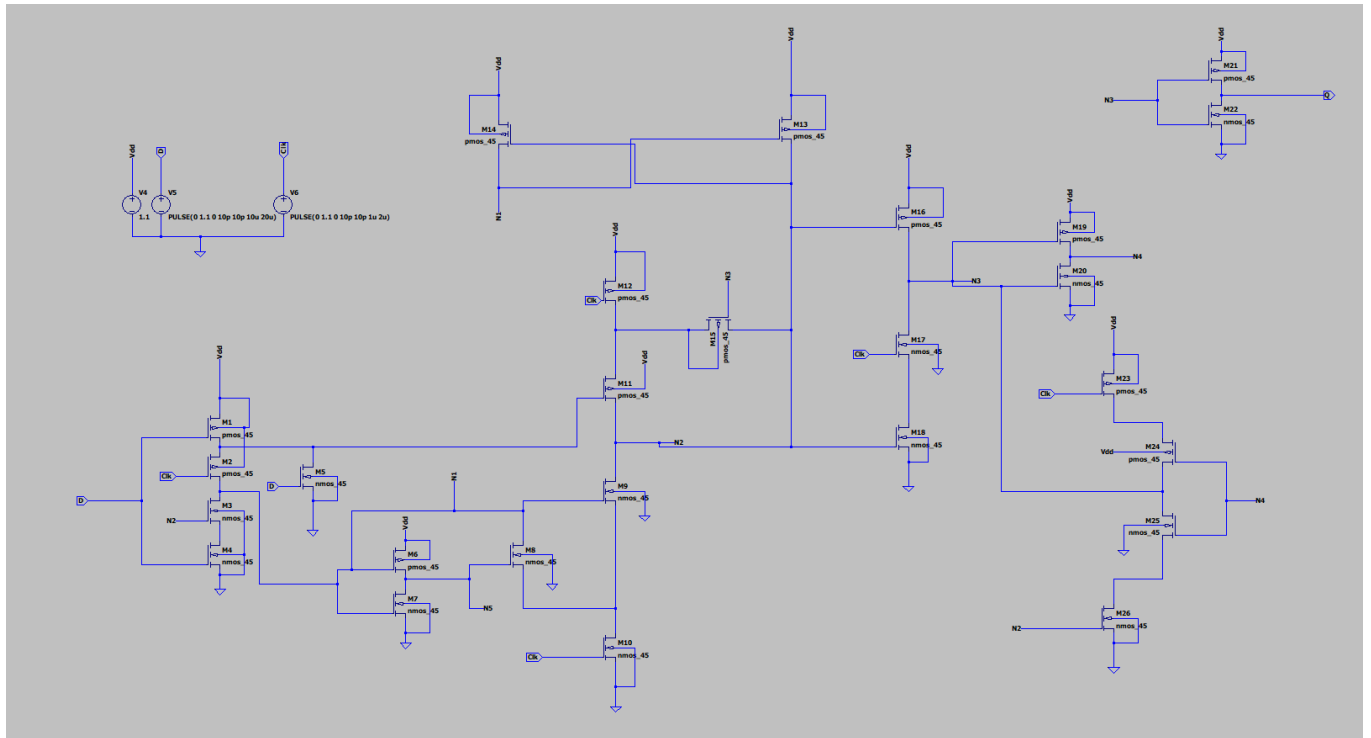
We are simulating the output Q by giving a clock and D as input from LtSpice for the proposed architecture. With the employment of input-aware precharge scheme, the proposed TSPC FF precharges only when necessary

## 1.1 SCHEMATIC

### 1.1.1 CADENCE VIRTUOSO

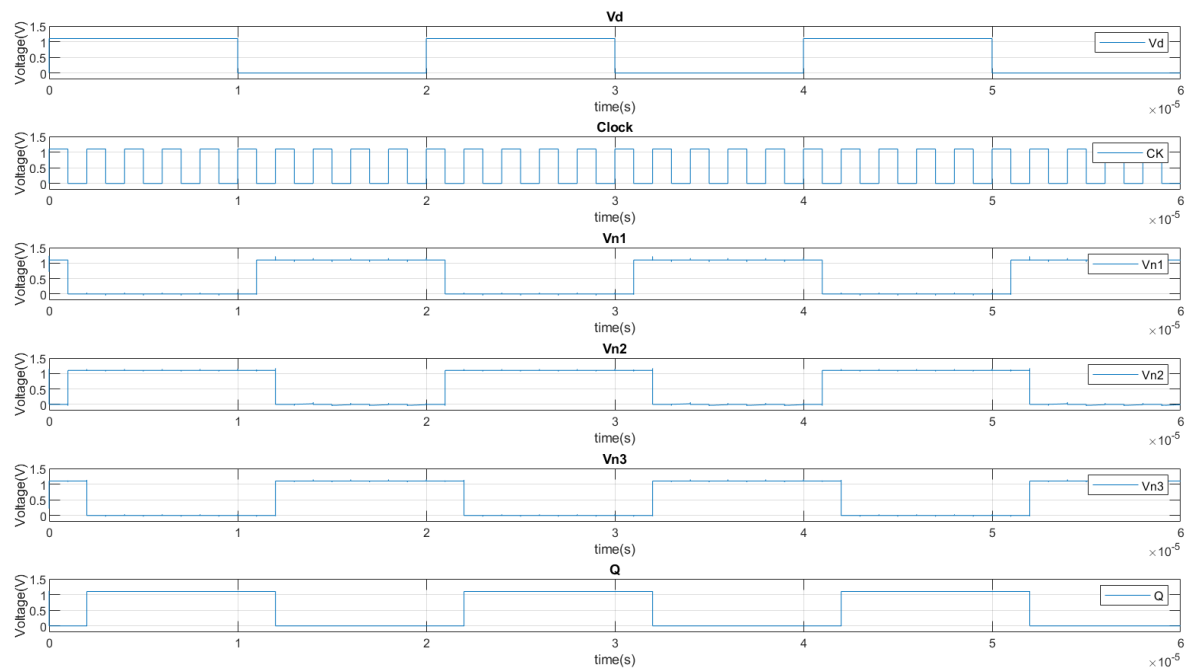


## 1.1.2 LTSPICE

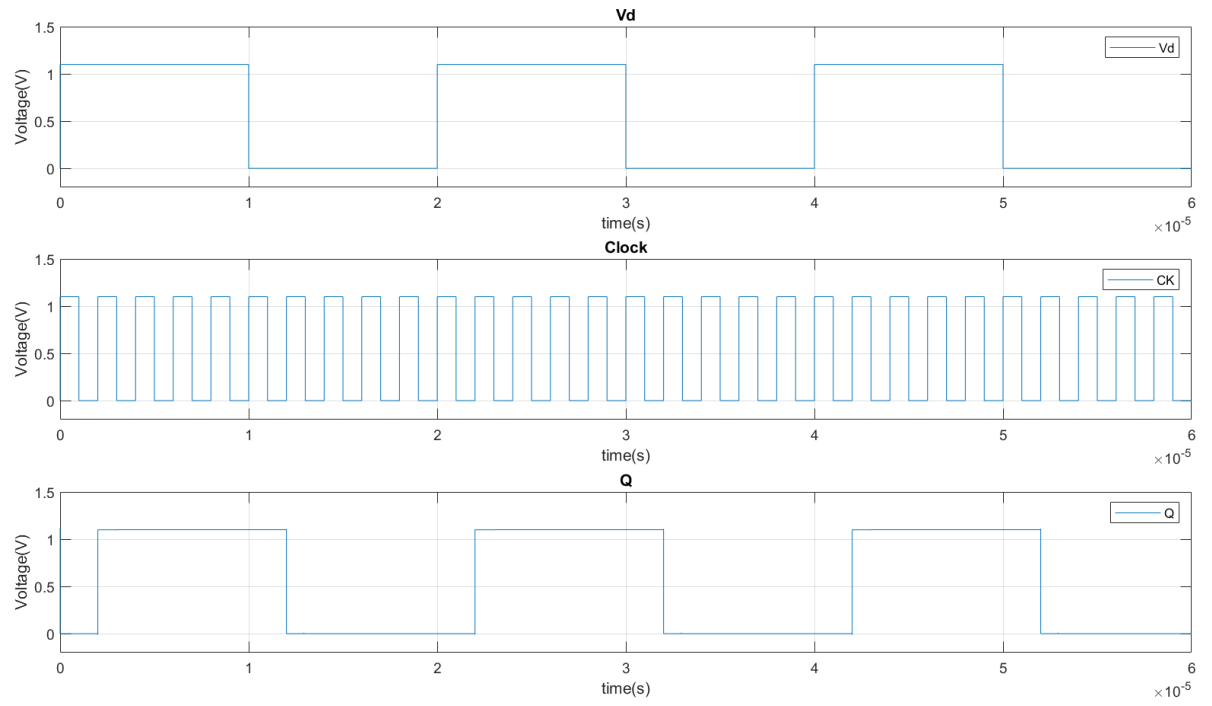


## 1.2 PLOTS

### 1.2.1



### 1.2.2

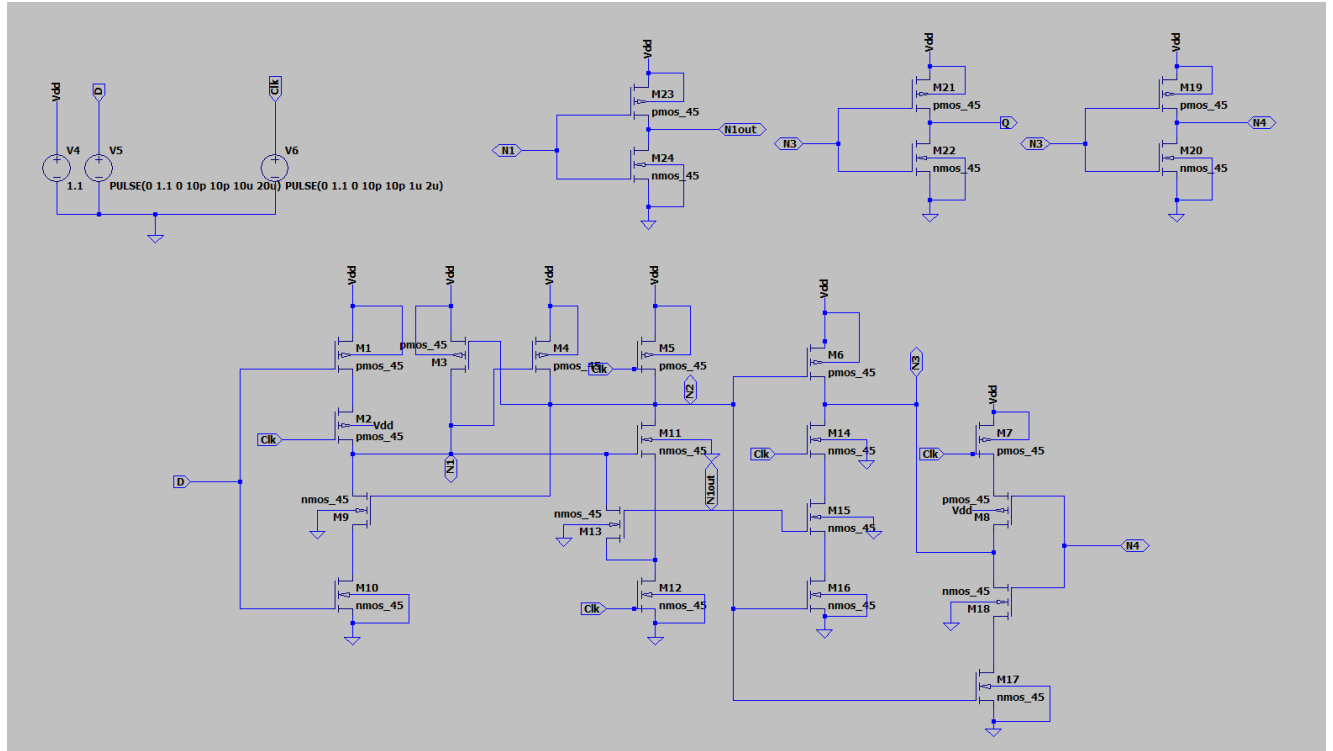


## 2 SSCFF

We are simulating the output Q by giving a clock and D as input from LtSpice for the SSCFF architecture. It is Static Single-Phase Contention-free flip flop. The main drawback is redundant precharges because of lack of input-aware precharge.

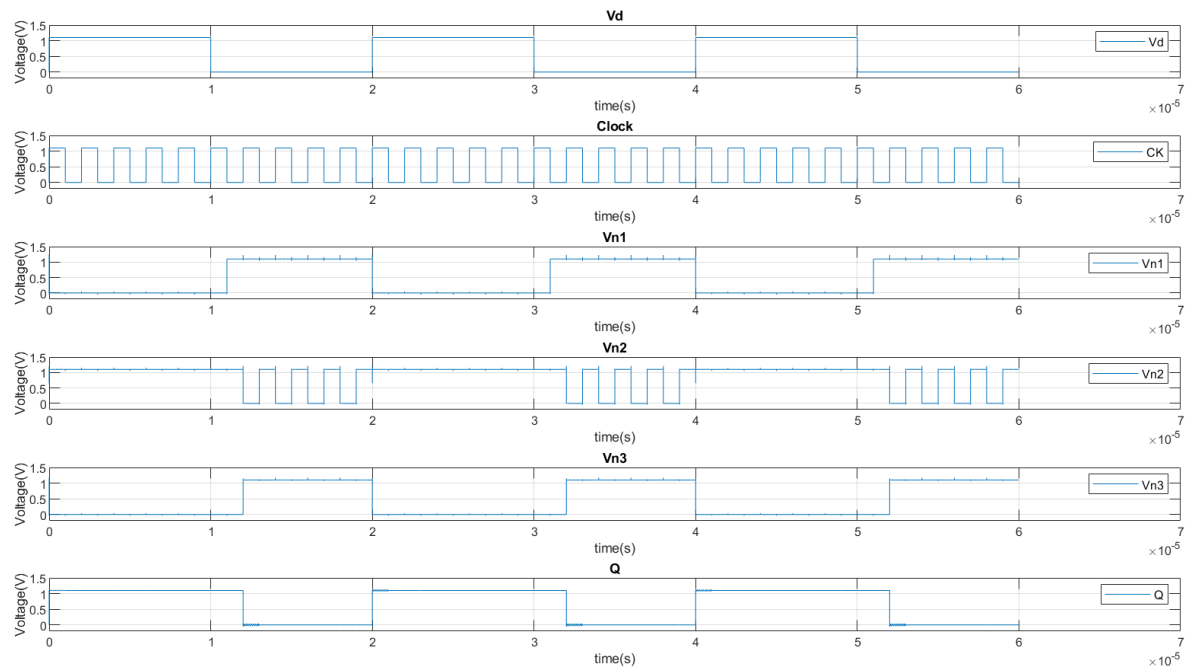
### 2.1 SCHEMATIC

#### 2.1.1 LTSPICE

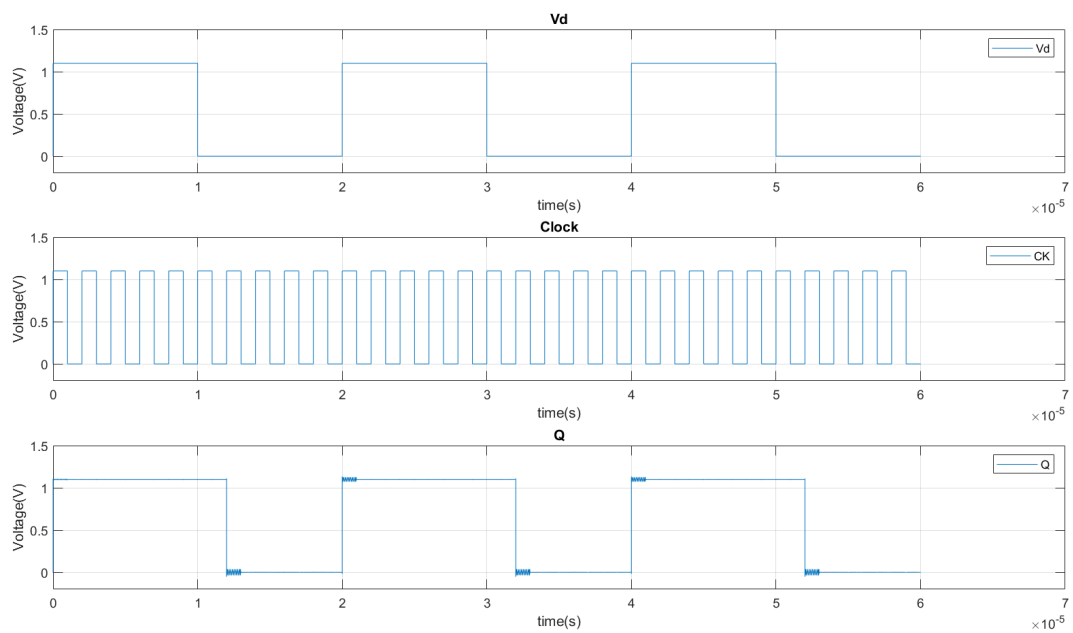


## 2.2 PLOTS

### 2.2.1



## 2.3

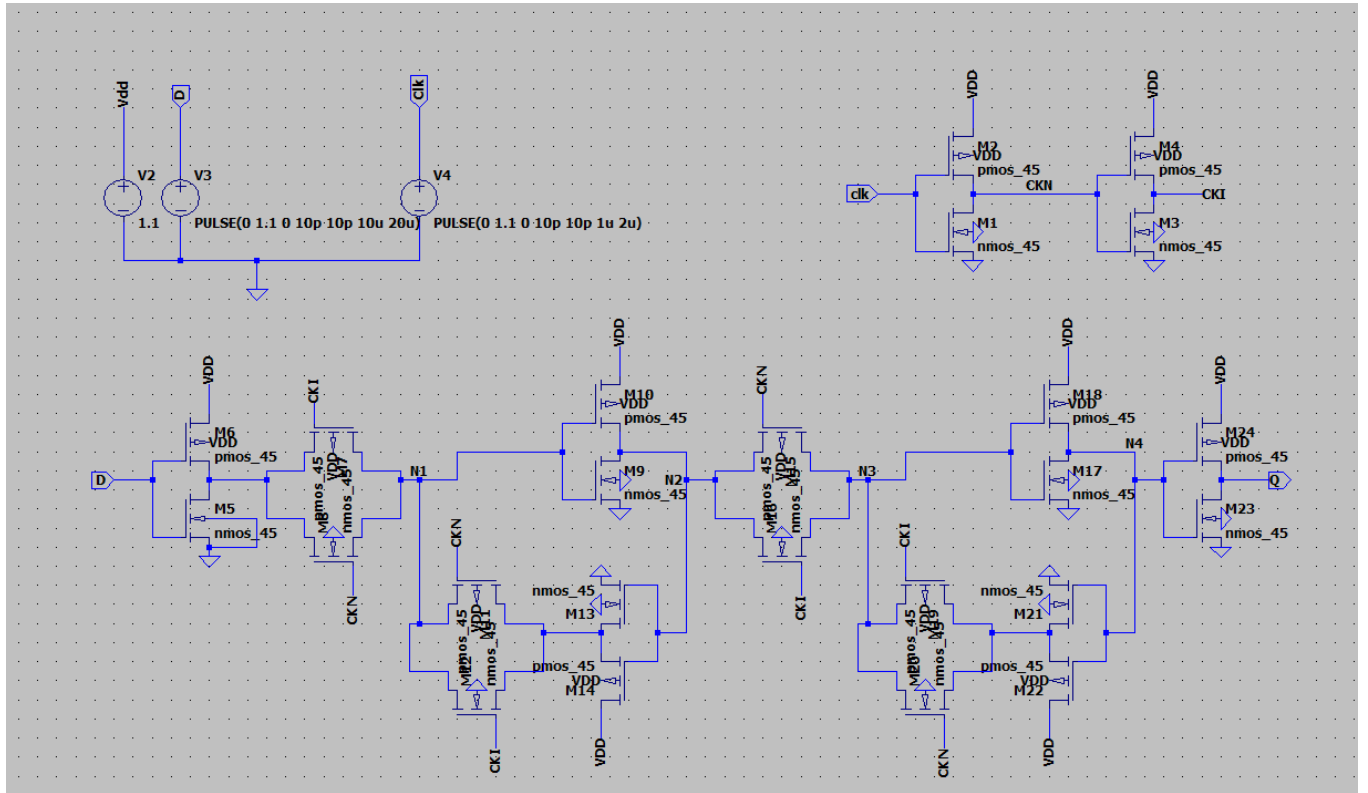


### 3 TGFF

The TGFF is a contention-free FF which is suitable for near-threshold operation. The main drawback of TGFF is the large clock network. The internal nodes CKN and CKI toggle no matter what the input data is, and the nodes CKN and CKI drive a larger number of transistors. Thus, the power consumption of TGFF is still large even if the data activity remains low.

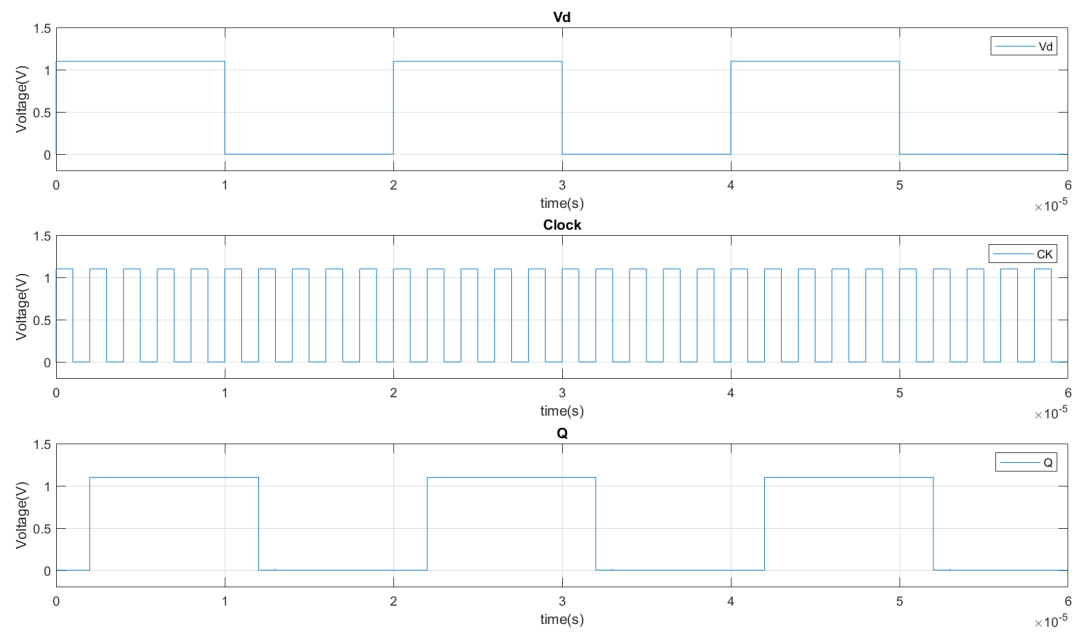
#### 3.1 SCHEMATIC

##### 3.1.1 LTSPICE





## 3.2 PLOTS



## 4 ANALYSIS

### 4.1 THEORY

The analysis was done with ptm low power 45nm technology in LtSpice. The delay from clock to Q was calculated by subtracting the time taken by Q to reach 50% of its maximum value by time taken by CK for the same. Average power was calculated by multiplying supply voltage and source current. The plots are for 10% data activity. The technology node for TGFF was not mentioned in the paper, so 45 nm was used for analysis. We used PSO algorithm for finding the optimal widths.

PSO Code used:

```
import os
from PyLTSpice.LTSpiceBatch import SimCommander
from PyLTSpice.LTSteps import LTSpiceLogReader
import numpy as np
import pyswarms as ps
from pyswarms.utils.functions import single_obj as fx

def LTSpice_PSO(W):
    pdp = [100.0]*10
    for i in range(10):
        LTC = SimCommander("proposed.asc")
        LTC.set_parameters(w1=W[i][0], w2=W[i][1], w3=W[i][2], w4=W[i][3], w5=W[i][4], w6=W[i][5], w7=W[i][6], w8=W[i][7], w9=W[i][8], w10=W[i][9])
        LTC.run()
        LTC.wait_completion()
        f = open("proposed.log", "r")
        data = f.read()
        x = data.split("\n")
        for j in range(len(x)):
            if x[j][0:6] == "delay:":
                print(x[j])
                pdp_str = x[j].split("=")
                pdp[i] = float(pdp_str[1])
                break
    return pdp

min_bound = np.ones(10)*90e-9
max_bound = np.ones(10)*240e-9
bounds = (min_bound, max_bound)
options = {'c1': 0.5, 'c2': 0.3, 'w': 0.9}
optimizer = ps.single.GlobalBestPSO(n_particles=10, dimensions=10, options=options, bounds=bounds)
cost, pos = optimizer.optimize(LTSpice_PSO, iters=30)
```

Results in the paper:

Delay (Clock - Q) - TGFF:104.73 ps; S2CFF:73.11 ps; Proposed:77.31 ps

Normalized Power - TGFF:1037.93 nW; S2CFF:360.79 nW; Proposed:162.18 nW

Obtained Results:

Delay (Clock - Q) - TGFF:144 ps; S2CFF:82.8 ps; Proposed:24.4 ps

Normalized Power - TGFF:645 nW; S2CFF:328 nW; Proposed:220 nW

## 4.2 PLOTS

### 4.2.1

```
Release Notes: 1.76.1  pso.py  TGFF.log  report.log  proposed.log  SpiceBatch.log
TGFF.log
12  Gmin = 1.0455e-009
13  Gmin = 1.76685e-007
14  Gmin = 1.89714e-008
15  Gmin = 2.03704e-009
16  Gmin = 2.18725e-010
17  Gmin = 2.34854e-011
18  Gmin = 2.52173e-012
19  Gmin = 2.70769e-013
20  Gmin = 0
21  Gmin stepping succeeded in finding the operating point.
22
23
24  t1: time=1.0005e-008 at 1.0005e-008
25  t2: time=1.01493e-008 at 1.01493e-008
26  clktoq: t2-t1=1.44258e-010
27  power: AVG(-v(vdd)*i(v2))=6.45518e-007 FROM 0 TO 2e-007
28  variable: power*clktoq=9.31211e-017
29  delay: t2-t1=1.44258e-010
30
31
32  Date: Tue Mar 14 16:49:28 2023
33  Total elapsed time: 0.377 seconds.
34
```

### 4.2.2

```
Release Notes: 1.76.1  pso.py  s2cff.log  report.log  proposed.log  SpiceBatch.log
s2cff.log
1  Circuit: * C:\sem6\VLSI_arch\project\s2cff.asc
2
3  Direct Newton iteration for .op point succeeded.
4
5  t1: time=1.00005e-007 at 1.00005e-007
6  t2: time=1.00088e-007 at 1.00088e-007
7  clktoq: t2-t1=8.28239e-011
8  power: AVG(-v(vdd)*i(v1))=3.28452e-007 FROM 0 TO 2e-007
9  variable: power*clktoq=2.72036e-017
10 delay: t2-t1=8.28239e-011
11
12
13 Date: Tue Mar 14 16:09:33 2023
14 Total elapsed time: 0.485 seconds.
15
16 tnom = 27
17 temp = 27
18 method = modified trap
19 totiter = 6611
20 traniter = 6571
21 tranpoints = 2290
22 accept = 1980
23 rejected = 310
24 matrix size = 115
25 fillins = 10
26 solver = Normal
```

### 4.2.3

```
Release Notes: 1.76.1  psu.py  report.log  proposed.log X  SpiceBatch.log
proposed.log
1  Circuit: * C:\sem6\VLSI_arch\project\proposed.asc
2
3  Direct Newton iteration for .op point succeeded.
4
5  t1: time=1.10005e-007 at 1.10005e-007
6  t2: time=1.10029e-007 at 1.10029e-007
7  clktoq: t2-t1=2.4433e-011
8  power: AVG(-v(vdd)*i(v1))=2.19952e-007 FROM 0 TO 2e-007
9  variable: power*clktoq=5.37409e-018
10 delay: t2-t1=2.4433e-011
11
12
13 Date: Tue Mar 14 15:21:27 2023
14 Total elapsed time: 0.282 seconds.
15
16 tnom = 27
17 temp = 27
18 method = modified trap
19 totiter = 6107
20 traniter = 6064
21 tranpoints = 2263
22 accept = 1955
23 rejected = 308
24 matrix size = 123
25 fillins = 4
26 solver = Normal
27 Thread vector: 25.3/19.8[2] 4.0/2.2[2] 1.1/0.9[2] 0.4/0.7[1] 2592/500
28 Matrix Compiler1: 18.79 KB object code size 2.8/1.3/[0.9]
29 Matrix Compiler2: 15.56 KB object code size 1.1/1.3/[0.6]
```

## 5 FUTURE WORK

We are simulating the mentioned architectures in cadence virtuoso and will obtain setup and holdtime. It will be compared for different data activity. We will plot Schematic of the proposed FF with (a) set, (b) reset, and (c) scan.